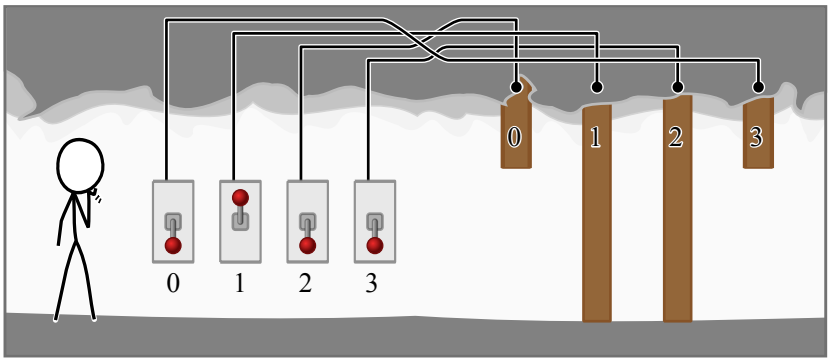
International Olympiad in Informatics 2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

cave

Netherlands — 1.0

Onderweg van je college naar de UQ Centre ben je tegen de de ingang van een verloren gewaand grottenstelsel aangelopen. Dit stelsel loopt diep onder de universiteit door.

De ingang wordt geblokkeerd door een beveiligingssysteem dat bestaat uit `N` opeenvolgende deuren en `N` schakelaars. Elke volgende deur staat achter de vorige. Elke schakelaar is met een andere deur verbonden.



De deuren zijn op volgorde genummerd: `0, 1, …, (N - 1)` waarbij deur 0 het dichtste bij je is.

De schakelaars zijn ook genummerd: `0, 1, …, (N - 1)`. Je weet echter niet welke schakelaar verbonden is met welke deur.

De schakelaars zitten bij de ingang van de grot. Elke schakelaar kan *omhoog* of *omlaag* staan. Voor elke schakelaar is er maar één correcte stand. Als een schakelaar in de juiste stand staat dan is de bijbehorende deur open. Als een schakelaar niet in de juiste stand staat dan is de bijbehorende deur gesloten. De juiste stand kan voor elke schakelaar anders zijn. Je weet niet wat de juiste stand van de schakelaars is.

Je wilt graag het beveiligingssysteem begrijpen.

Om dit te doen mag je de schakelaars in een willekeurige combinatie van standen zetten. Daarna kun je de grot inlopen om te zien wat de eerste gesloten deur is.

Je kunt niet door een gesloten deur kijken. Zodra je de eerste gesloten deur tegenkomt is niets te zien van de erachter gelegen deuren.

Je hebt slechts tijd om `70.000` combinaties van de schakelaars te proberen.

Jouw opdracht is om te bepalen wat de correcte stand van elke schakelaar is en ook welke schakelaar met welke deur is verbonden.

Implementatie

Stuur een bestand in dat de volgende procedure implementeert: `exploreCave()`. Deze procedure mag de volgende functie van de grader: `tryCombination()` maximaal `70.000` keer aanroepen. Je moet eindigen met een aanroep van grader procedure: `answer()`. Deze functie en de procedures worden hieronder beschreven.

Grader Functie: `tryCombination()`

C/C++
```
int tryCombination(int S[]);
```

Pascal
```
function tryCombination(var S: array of LongInt) : LongInt;
```

Beschrijving

Deze functie is door de grader geimplementeerd.

Deze functie laat je een combinatie van schakelaars proberen. Je kunt er dan achter komen wat de eerste gesloten deur is.

Als alle deuren open zijn dan levert de functie `-1` als resultaat.

Deze functie is van orde `O(N)`; de runtime van de aanroep is in het slechtste geval recht evenredig met N.

Je mag deze functie maximaal `70.000` keer aanroepen.

Parameters

- `S`: Een array met lengte `N`, waarin de stand van elke schakelaar staat aangegeven. Het element `S[i]` geeft de stand van schakelaar `i` aan. De waarde `0` geeft aan dat de schakelaar omhoog staat. De waarde `1` geeft aan dat de schakelaar omlaag staat.
- *Resultaat*: Het nummer van de eerste gesloten deur, of `-1` als alle deuren open staan.

Grader Procedure: `answer()`

C/C++
```
void answer(int S[], int D[]);
```

Pascal
```
procedure answer(var S, D: array of LongInt);
```

Beschrijving

Roep deze procedure aan als je hebt ontdekt wat de juiste combinatie van schakelaars is om alle deuren open te hebben, en de deur waarmee elk van de schakelaars verbonden is.

Parameters

- `S`: Een array van lengte `N`, dat aangeeft wat de juiste stand van elke schakelaar is. Het format is gelijk aan wat bij de functie `tryCombination()` beschreven is.
- `D`: Een array van lengte `N`, dat aangeeft welke deur bij welke schakelaar hoort. Element `D[i]` moet het nummer van de deur bevatten waar schakelaar `i` mee verbonden is.
- *Resultaat*: Je krijgt geen resultaat. Deze procedure beëindigt je programma.

Jouw Procedure: `exploreCave()`

C/C++
```
void exploreCave(int N);
```

Pascal
```
procedure exploreCave(N: longint);
```

Beschrijving

Jouw inzending moet deze procedure implementeren.

Deze functie moet de grader routine `tryCombination()` gebruiken om te bepalen wat de juiste stand van elke schakelaar is, en welke deur bij welke schakelaar hoort. Deze functie moet `answer()` aanroepen zodra je dit bepaald hebt.

Parameters

- `N`: Het aantal schakelaars en deuren in de grot.

Voorbeeld

Stel dat de deuren en schakelaars verbonden zijn zoals in het plaatje hierboven.

Functie aanroep	Resultaat	Uitleg
`tryCombination([1, 0, 1, 1])`	`1`	Dit komt overeen met het plaatje. Schakelaars 0, 2 en 3 staan omlaag; schakelaar 1 staat omhoog. De functie geeft als resultaat `1` om aan te te geven dat deur 1 de eerste gesloten deur is die je tegenkomt.
`tryCombination([0, 1, 1, 0])`	`3`	De deuren 0, 1 en 2 zijn open; deur 3 is gesloten.
`tryCombination([1, 1, 1, 0])`	`-1`	Door schakelaar 0 ook omlaag te zetten gaan alle deuren open. Dit zie je aan het resultaat van `-1`.
`answer([1, 1, 1, 0], [3, 1, 0, 2])`	*(En klaar)*	We denken dat de correcte combinatie `[1, 1, 1, 0]` is, en dat de schakelaars 0, 1, 2 en 3 verbonden zijn met respectievelijk deur 3, 1, 0 en 2.

Randvoorwaarden

- Tijdslimiet: 2 seconden
- Geheugenlimiet: 32 Mb
- $1 \le N \le 5.000$

Subtasks

Subtask	Punten	Aanvullende randvoorwaarden
1	12	Schakelaar `i` is steeds verbonden met deur `i`. Je hoeft alleen te bepalen wat de juiste combinatie van schakelaars is.
2	13	De juiste combinatie is altijd `[0, 0, 0, ..., 0]`. Je hoeft alleen te bepalen welke deur bij welke schakelaar hoort.
3	21	$N \le 100$
4	30	$N \le 2.000$
5	24	*(geen)*

Stoeien

De voorbeeldgrader leest invoer uit `cave.in`. Dit bestand moet in het volgende format zijn:

- regel 1: `N`
- regel 2: `S[0] S[1] … S[N - 1]`
- regel 3: `D[0] D[1] … D[N - 1]`

`N` is het aantal deuren en schakelaars `S[i]` is de juiste stand voor schakelaar `i`, en `D[i]` is de deur waarmee schakelaar `i` verbonden is.

Het bovenstaande voorbeeld zou je in het volgende format moeten weergeven:

```
4
1 1 1 0
3 1 0 2
```

Language Notes

C/C++	Je moet doen: `#include "cave.h"`.
Pascal	Definieer `unit Cave`, en importeer de routines van de grader middels `uses GraderHelpLib`. De nummering van arrays begint met `0` (niet `1`).

Op je computer staan templates van de oplossingen. Bekijk die voor voorbeelden.